An OCR Concept for Historic Prints

Ursina Caluori and Klaus Simon
Swiss Federal Laboratories for Materials Science and Technology (EMPA); Dübendorf, Switzerland

Abstract

We present a new concept for the recognition of single characters, the core component of every OCR-System. The recognition is organized as pattern matching with on-the-fly generated patterns corresponding to the glyphs of a given computer font. Thereby, the set of currently considered fonts can be chosen and exchanged arbitrarily which allows a flexible adaption of the software to each specific task, in particular the needs of historic prints.

Motivation

Recently, mass-digitization of historic prints has become a popular issue for libraries. For instance, in the *European Digital Library* program the EU strives for renewing Europe's printed heritage as digitally available resources where the final challenge is *transforming digital images of scanned books into electronic text*. However, there are serious technological challenges. For instance, on the website of IMPACT[1], one of the corresponding EU-projects, the following statement can be found:

> *Automated text recognition, carried out by Optical Character Recognition (OCR) engines does in many cases not produce satisfying results for historical documents. Recognition rates are poor or even useless. No commercial or other OCR-engine is able to cope satisfactorily with the wide range of printed materials published between the start of the Gutenberg age in the 15th century and the start of the industrial production of books in the middle of the 19th century.*

In absence of alternatives, the usual storage of digital scans as image files turns the used scan resolution into a fundamental limiting factor in mass-digitizations. For that reason, libraries often accept low scan resolutions and high image compression rates, resulting in scans that are hard to read for humans and OCR-programs alike. Consequently, OCR-applications for historic prints show in many cases poor error rates, far away from the usual $1-2\,\%$ of modern office applications.

However, suboptimal scan quality is only one reason for insufficient OCR-results. Another reason may be the classification concept of modern OCR-software, for instance *neural networks* or *support vector machines*, see [3, 5, 7, 8] for a survey, which follows the pattern recognition approach. Here we have to distinguish between *pattern recognition* and *pattern matching*, where Wikipedia explains [2]

> *Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to do* fuzzy *matching of inputs. This is opposed to pattern matching algorithms, which look for exact matches in the input with pre-existing patterns.*

In other words, modern OCR-software tries to recognize the *common* or *average* shape of a character, for example "R", in contrast to the specific graphical appearance of this character in a font like Helvetica, e.g."R". Thereby, the "common shape" is determined by a set of representative examples which have to be learned in a supervised training process during the development phase of the software. The central advantage of this concept is generality, i.e. the concrete appearance of a character may vary in a certain range which is, for instance, significant for the recognition of handwritings. The drawbacks are on the one hand inflexibility – exchange of the training set requires the development of a new software – and on the other hand principal limited recognition accuracy since the training is related to an average.

Our concept is based on the pattern matching approach. We compare a *sample* with a set of glyphs where a *glyph* denotes the specific graphical shape of a character in a font. The comparison between the sample and a glyph results in a similarity mass, e.g. the number of common pixels. In principal, our classification process determines the most similar glyph related to the given sample. The advantage of this procedure is its flexibility. The approach works for arbitrary fonts. Therefore, the fonts currently needed for given task can be embedded in the scan data. Furthermore, the necessary font technology is very well established, is highly efficient and widely available as a common part of every publishing software. On the other hand, our approach needs a good scan quality and a sufficiently high resolution.

In chapter 2 we describe our approach in greater detail. Next, we consider a recognition test for glyphs in order to show that our concept is realistic. Finally, we analyze the state of our development.

An Alternative for Glyph Recognition

A typical OCR-workflow [3, 5] starts with the scanning of the original. The captured appearance of a scanned glyph differs from its definition in a corresponding font description both by systematic and stochastic reasons. Accordingly, usually a preparation step follows, including several improvements and regularizations of the input image. Then, every pixel is rounded to 0 (black) or 1 (white) where *white* stands for the background and *black* means a dot belonging to some character. The next step, called *segmentation*, groups adjacent dots into *segments* where segments or groups of segments correspond to the captured appearance of the glyphs. Then a set of weighing functions, called *features*, is applied to each segment. The features form a *feature vector* which represents the segment in an n-dimensional *feature space*. The core task of an OCR-system is the assignment of a fitting glyph to each segment, called *classification*. Popular under-

[1] http://www.impact-project.eu

[2] cited from http://en.wikipedia.org/wiki/Pattern_recognition

standings of classification consist in separating the feature space into distinct regions which are in accordance with the characters to recognize. However, there are a lot of different approaches, see for instance [7]. Finally, the plausibility of the classification is checked by linguistic methods.

Our Concept

There is a very simple and obvious way to carry out the classification. Assuming that the underlying text was produced with a font F, we can compare a segment with every glyph of F and choose the best match. Hardly surprising, the historically first OCR-concepts were based on this procedure, well-known as *pattern matching*. However, this requires that the font F or, more accurately, the corresponding glyphs are known to the software. Accordingly, OCR-projects from the seventies used fixed, specially designed fonts, for instance the famous OCR-A (1968) and OCR-B (1972) fonts. Later, with more complex challenges like the recognition of handwriting, the pattern matching approach became unpopular in the nineties. Obviously, generality is more important for changing requirements in an office environment than accuracy.

With this paper we wanted to motivate the renewal of pattern matching for certain OCR-applications. Please note that the situation in mass-digitization is different from an office environment. For instance, the layout of a newspaper is more or less stable over a certain period of time such that the same fonts and glyphs, respectively, appear on potentially tens of thousands of pages. In other words, in a mass-digitization application it is very natural and reasonable to adapt and prepare an OCR-software to a specific task by communicating involved fonts and font sizes to the software.

The question now is how to realize this purpose technically. On second thought, it seems obvious that the technology currently used to produce such a layout[3], should be able to specify fonts and glyphs, respectively, for an OCR-software. Indeed, there is no problem to integrate a modern font rasterization engine into an OCR-software, enabling it to render text onto bitmaps. The resulting glyph images are of the same kind as the scan images and can therefore be used to carry out a pattern matching without any further difficulties.

The integration of a rasterization engine — we use the *Free Type 2* [10] font library for C++ — enables us to do pattern matching on a new level. The considered patterns are specified by the embedded fonts in the scan job and can be generated spontaneously when needed at any pt-size. The high efficiency of Free Type 2 allows the generation of many thousands of patterns in less than a second.

Implementation

Let us now turn to the implementation aspects. The segmentation is carried out in the same way for the scan image and glyph images. Thereby, we concentrate consecutive black pixels to horizontal intervals. The resulting run length encoding of segments and patterns is both space- and time-efficient. Moreover, the pattern matching is based on this interval structure. Some additional information is maintained for every segment: The number of pixels and intervals, the size and coordinates of the bounding box, the coordinates of the center of gravity and so on.

A pattern is formed by segments found in its glyph image together with additional information about its font, its character and its pt-size. Please note that a pattern can consist of multiple segments, like for example an “i”. For that reason, we sometimes group segments of the scan image together. Accordingly, a *sample* always means a combination of one or more segments to be classified as a character.

The pattern matching compares a sample with a pattern and computes a similarity mass between them. In general, the sample S and a pattern P differ in their bounding boxes. In order to find the best superposition we compare the outlines of S and P. Let a_i be the distance from the left bounding box border of P to the first pixel in line i. Then $a_1, \ldots, a_t$ forms the left outline of P. The other outlines of P and the outlines of S are defined in a similar way. The vertical superposition of S and P is determined by the maximum cross correlation between their left and right outlines. The determination of horizontal superposition is analog.

To accomplish the pattern matching we compare the pixel lines of S with those of P. Let A and B be such a pair of lines. Both A and B consist of lists of intervals.[4] The intersection $A \cap B$ forms a new list of intervals in the same way as the corresponding symmetrical difference does. Let now $\{D_1, \ldots, D_d\}$ be the union of all intervals of all line intersections and let $\{U_1, \ldots, U_u\}$ be the analog for the symmetrical difference. Then we define our similarity mass $M(S,P)$ between S and P as

$$M(S,P) \;=\; \frac{\sum_{i=1}^{d} (2 \cdot |D_i|)^2}{\sum_{i=1}^{d} (2 \cdot |D_i|)^2 \;+\; \sum_{i=1}^{u} |U_i|^2} \qquad (1)$$

where $|D_i|$ stands for the length of D_i. Please note that the computation of (1) can be done in the number of intervals in S and P.

Although a rasterization engine supports arbitrary scalings of glyphs it is neither usual nor reasonable to tap the full potential. Historic prints are produced in metal types which were only available in a fixed number of font sizes, for instance, 10 pt as normal size, 9 pt for footnotes and 14 pt for titles. The desktop publishing does not have this physical restriction of fixed font sizes but it is an established typographical tradition to do so. Since we use the same technology we adapt this tradition, i.e. the user of our OCR-software has to specify a set of font sizes as input parameter to our program. This reduces the number of necessary pattern generations and speeds up the subsequent processes.

In principle, our classification is a search process over the considered patterns. Therefore, the running time of our software can be optimized in a classical algorithmic manner. However, in order to keep our performance test in the next chapter as simple as possible we use here only a size check to compare the bounding boxes.

Testing the Glyph Recognition

We designed some documents each consisting of several lines with the letter-figure-combination:

a, b, …, z, 0, 1, …, 9

[3] namely any kind of desktop publishing software

[4] formed by consecutive dots

	lowercase		uppercase	
Font	300	600	300	600
Arial	98.15	98.84	98.38	97.69
Bauer Bodoni bold	96.30	97.22	98.15	98.61
Breitkopf Fraktur	97.92	98.84	98.38	98.38
Computer Modern Roman	99.07	99.07	99.07	99.07
Courier New	98.15	100	98.38	100
Courier New bold	99.77	100	98.84	100
Gill Sans	97.92	98.15	97.92	99.31
Helvetica	98.38	99.31	98.61	99.77
Myriad Pro	98.84	99.31	99.31	100
Stone Serif bold	99.54	100	97.22	99.77
Times New Roman	98.61	99.54	98.38	98.84
Times New Roman bold	98.61	98.84	99.54	99.54
Unger Fraktur	97.22	99.07	98.15	98.84
Verdana	99.31	100	99.54	99.77
Walbaum Fraktur	96.53	98.15	98.84	99.31
total	98.29	99.09	98.58	99.26

Table 1: Recognition rate (%) by font in 300 and 600 ppi

	lowercase		uppercase	
Font Size	300	600	300	600
6 pt	94.07	96.48	96.85	97.04
7 pt	96.85	97.22	97.22	98.70
8 pt	98.15	98.52	97.41	99.07
9 pt	97.22	98.15	96.67	99.07
10 pt	97.59	99.63	98.70	99.44
11 pt	97.04	99.44	97.78	98.89
12 pt	99.07	99.63	99.44	99.81
14.4 pt	99.63	100	100	99.81
17.28 pt	100	100	99.63	99.81
20.74 pt	100	100	99.81	99.81
24.88 pt	99.81	100	99.81	99.81
27 pt	100	100	99.63	99.81

Table 2: Recognition rate (%) by font size in 300 and 600 ppi

The lines are typeset in the font sizes 6 pt, 7 pt, ..., 12 pt, 14.4 pt, 17.28 pt, 20.74 pt, 24.88 pt and 27.0 pt. Each document was typeset in one of the fonts: Arial, Bauer Bodoni bold, Breitkopf Fraktur, Computer Modern Roman, Courier New, Courier New bold, Gill Sans, Helvetica, Myriad Pro, Stone Serif bold, Times New Roman, Times New Roman bold, Unger Fraktur, Verdana and Walbaum Fraktur.[5] For the capitals we created corresponding documents. These documents are then printed with a Xerox Phaser 7500 laser printer and scanned at 300 and 600 ppi with an Epson Perfection V700 Photo scanner. The resulting gray scale images are then used as input files to our software.

The recognition test was carried out with a reduced version of our software. The search for the most similar pattern was split into two phases. A first rough estimation uses a quick, simple size check of bounding boxes. If the first phase found a pattern with a similarity bigger than 90 % the search stops. Otherwise a second search without any constraints is launched. This is a primitive version of a branch and bound search heuristic and is typical for our running time optimizations. The considered patterns are given by the corresponding font family, as far as this family was available to us. Usually a family consists of a regular, a bold, an italic and a bold italic font such that we considered about 1000 glyphs each in 12 different font sizes.

The recognition result was evaluated and recorded automatically in form of both an ASCII-text output and a graphic visualization where correct classifications are indicated in green and wrong ones in red, see figures 1 and 2. The set of all input and output images can be found on our website: `http://empamedia.ethz.ch`

Results

For 300 ppi we get an overall recognition rate of 98.29 % for lowercase letters (and numbers) and 98.58 % for the capitals. For 600 ppi we get 99.09 % and 99.26 %. Table 1 contains the corresponding values for each font, table 2 for each font size, respectively.

The worst case is Bauer Bodoni bold with 96.30 % in lowercase letters at 300 ppi. Possibly, this is correlated with the extreme line thickness differences well-known for this font.

As expected, the recognition rate increases with scan resolution and font size. The very good result for large font sizes lets us suppose that there is no principle limit for the accuracy of our method. For the most relevant sizes around 10 pt we find that doubling the resolution at least halves the error rates. Please note that we used normal office equipment in our workflow. At the lower end at 6 pt and 300 ppi the scan process came to its limits. The letters began to crumble, thin lines and small details broke into pieces or simply disappeared. Surprisingly, our algorithm still yields a recognition rate of 94.07 % for all fonts. On the other hand, already 8 pt leads to 98.15 % recognition rate for the same resolution.

The observed errors can roughly be divided into two groups. The first one is correlated with graphically very similar patterns. In some of the fonts an "ℓ" is identical or nearly identical to a "1" or an "I". Since our software only estimates the graphical similarity, it could not discriminate between such cases. Nearly all

[5] The encoding of fraktur fonts differs a little bit from the others. Hence some characters in the fraktur documents are different.

Figure 1. *Result for Times New Roman lowercase letters and numbers at 600 ppi*

Figure 2. *Result for Unger Fraktur uppercase letters at 600 ppi*

errors at higher font sizes fall into this category. The second error type is due to the size check applied to the bounding boxes. For large differences the subsequent similarity check is left out. This saves a lot of running time since computing the similarity mass is much more expensive than checking the bounding boxes. However, sometimes we pass over the correct pattern and can thus only give a suboptimal answer. In state of the art OCR-software both types of errors are usually detected in context-related plausibility checks. Obviously, a glyph recognition cannot solve the OCR-problem alone. However, a wrong but similar classification is a good start for a subsequent correction.

Current State and Outlook

Our performance test shows the high potential of combining pattern matching with font technology for OCR-applications. The concept works more or less for every considered font but, of course, not identically for all of them. The fraktur fonts show no significant differences to sans serif or classical antiqua fonts. For this reason, we expect that our approach is especially useful for the recognition of historic prints.

In this paper, we considered a very reduced implementation in order to analyze the pure principle. Although the results so far are quite compelling, the concept can be improved in many ways.

As mentioned above, errors often happen between graphically similar or even identical patterns. In such cases the discrimination with a similarity mass is mostly guessing. It would be more reasonable to put identical or nearly identical glyphs into an equivalency class and replace them by a common representative. Their discrimination should then be postponed to subsequent linguistic checks. On the other hand, we frequently mix up "e" with "c" which means that we have an overall high similarity but clear differences in special regions. These typical differences can be checked additionally whenever a sample is categorized as an "e" or "c". These kinds of special constraints are easy to implement and improve the performance significantly.

The central aspect of our approach is the pattern matching with on-the-fly generated patterns. Up to now, only glyph images are used as patterns, but this does not mark the end of possibilities. Quite the contrary, parts or modifications of glyph images can be involved too, allowing us to handle degenerations of samples, e.g. if the point in "i" is missing. Another nice option is the possibility to adapt a pattern on-the-fly to the appearance of already observed samples. If the observed distortions have a systematic form, for instance a filter, then this filter can be applied to subsequent glyph images in order to reduce the gap between samples and patterns.

The maximum similarity search copes with different fonts and varying font sizes quite well. In particular, there is no supervised training necessary to apply a new font. Apart from this, some special adaptions to a given font may be useful. For instance, the average coat thickness of a font has an evident influence on our similarity mass which means that the 90 %-rule in phase 1 of the search process has to be adapted accordingly.

Our obvious application limitations are the scan quality and/or the resolution for small font sizes. It can be expected that with an additional algorithmic effort these limits can be improved. An established method to increase the robustness in such cases is the use of several independent similarity masses and majority decisions.

The greatest challenge for the further development is the optimization of the running time. For mass-digitizations the OCR-software has to be as fast as possible. Our branch-and-bound heuristic describes the typical problem: Looking for the best tradeoff between additional error caused by the search bounds and the speed up of the software. Several algorithmic disciplines are involved: computational geometry [1], graph exploration [4, 2, 9], constraint programming [6] and so on. Software optimization is always a difficult topic. However, the concept is designed for it and hence there may be a chance for a non-trivial result.

Conclusion

The core functionality of every OCR-concept is the glyph recognition. Based on this, we suppose that the glyph recognition is also a main problem in OCR-applications for historic prints. In this paper, we consider a new concept for glyph recognition realized as pattern matching where the patterns are generated on-the-fly with a font rasterization engine. The integration of modern font technology into an OCR-software induces the same advantages for the OCR-functionality as it had 20 years ago for the publishing software, in particular the arbitrary exchange of fonts, unlimited scaling, high efficiency and so on. Our performance test should be understood as a first hint that the new approach performs at least as well as the established concepts. However, there is still a lot of work to do.

References

[1] M. de Berg, O. Cheong, M. van Kreveld and M. Overmars, Computational Geometry: Algorithms and Applications, 3rd ed., Springer-Verlag, Heidelberg, 2008.

[2] S. Even, Graph Algorithms, Computer Science Press, 1979.

[3] H. Bunke and P. Wang, Handbook of Character Recognition and Document Image Analysis, World Scientific, Singapore/New Jersey/London/Hong Kong, 1997.

[4] T. Hu, Combinatorial Algorithms, Addision-Wesley, 1982.

[5] M. Cheriet, N. Kharma, C. Liu and C. Suen, Character Recognition Systems: A Guide for Students and Practitioners, Wiley-Interscience, 2007.

[6] K. Marriott and P. Stuckey, Programming with Constraints, The MIT Press, Cambridge, Massachusetts 02142, 1998.

[7] R. Duda, P. Hart and D. Stork, Pattern Classification, 2nd ed., John Wiley, New York, 2001.

[8] S. Rice, G. Nagy and T. Nartker, Optical Character Recognition: An Illustrated Guide to the Frontier, Kluwer Academic Publishers, Boston, 1999.

[9] R. Tarjan, Data Structures and Network Algorithms, Society for Industrial and Applied Mathematics Philadelphia, PA, 1983.

[10] Freetype 2, `http://www.freetype.org`.

Author Biography

Ursina Caluori studied Computer Science at ETH Zurich with specialization in computer graphics. She wrote her master's thesis about color management workflows in the EMPA media technology group. After receiving her master's degree in early 2008, she continued working at EMPA in the field of OCR, gamut mapping and color management..